RoentDek
Handels GmbH

Supersonic Gas Jets
Detection Techniques
Data Acquisition Systems
Multifragment Imaging Systems

TDC8HQ

User Guide



www.roentdek.com

TDC8HQ

目次

1 はじめに 5

1.1 特長 5

1.2 用途 5

2 ハードウェア 6

2.1 PCIe ボードの取り付け 6

2.2 TDC8HQ の入力およびコネクタ 6

2.3 ステータス **LED** 9

2.4 複数ボードの同期 9

2.4.1 複数ボードの接続 9

2.4.2 ClockBox 10

2.4.3 複数ボード用クレート 10

3 TDC8HQ の機能 11

3.1 グルーピングとイベント 11

3.2 オートトリガー・ファンクションジェネレータ 12

3.3 タイミングジェネレータ（**TiGer**） 12

3.3.1 トリガーソース 13

3.3.2 TiGer の例：**200 kHz** スタートパルスの生成 15

3.3.3 TiGer の例：複数ソースからの遅延出力 15

3.3.4 TiGer による **ADC** のトリガー 15

3.4 ゲーティング 16

3.5 トリガー可能 **ADC** 17

4 ドライバープログラミング **API** 18

4.1 定数 18

4.2 ドライバー情報 18

4.3 初期化 19

4.3.1 構造体 **xhptdc8_manager_init_parameters** 19

4.4 ステータス情報 20

4.4.1 情報取得用関数 20
4.4.2 Structure xhptdc8_static_info 21
4.4.3 構造体 xhptdc8_param_info 22
4.4.4 構造体 **xhptdc8_fast_info** 23
4.4.5 構造体 **crono_pcie_info** 23
4.4.6 Structure xhptdc8_temperature_info 24
4.4.7 構造体 **xhptdc8_clock_info** 25
4.5 設定 25
4.5.1 YAML 設定ファイル 26
4.5.2 構造体 **xhptdc8_manager_configuration** 26
4.5.3 構造体 **xhptdc8_device_configuration** 26
4.5.4 構造体 **xhptdc8_trigger** 28
4.5.5 構造体 **xhptdc8_tiger_block** 29
4.5.6 構造体 **xhptdc8_channel** 30
4.5.7 構造体 **xhptdc8_adc_channel** 30
4.5.8 構造体 **xhptdc8_grouping_configuration** 31
4.6 ユーザーデータストレージ 32
5 実行時制御(ランタイム コントロール) 33
5.1 ドライバーのステータス制御 33
5.2 データ読み出し 33
6 出力データフォーマット 34
6.1 メモリ管理 34
6.1.1 パケットの肯定応答 34
6.1.2 TDC8HQ 内部のメモリ レイアウト 34
6.2 構造体 **TDCHit** 34
6.2.1 TDCHit タイプ 35
7 テクニカルデータ 37
7.1 パフォーマンス 37
7.1.1 TDC 測定特性 37

7.1.2 タイムベース 37

7.1.3 ADC 37

7.2 電気的特性 38

7.2.1 電源 38

7.2.2 TDC 入力 38

7.2.3 ADC 入力 39

7.2.4 クロック入力 **J2** 39

7.3 DIN EN 61010-1 に基づく必要情報 40

7.3.1 製造元 40

7.3.2 使用目的およびシステムの組み込み 40

7.3.3 保管に関する環境条件 40

7.3.4 動作に関する環境条件 41

7.3.5 冷却 41

7.3.6 リサイクル 41

1 はじめに

TDC8HQ はストリーミング型の高解像度 Time-to-Digital コンバータです。デジタルパルスの立ち上がりまたは立ち下がりエッジの到達時間が、タイムスタンプの無限ストリームとして記録されます。コモンスタートまたはコモンストップ動作をエミュレートするグルーピングモードも利用可能です。

1.1 特長

- 8 ピコ秒解像度を備えた 8 チャンネル・ストリーミング TDC
- ビンサイズ：13 ps
- ダブルパルス解像度：5 ns
- 読み出し用デッドタイム：なし
- L0 FIFO：15 ワード/チャンネル
- L1 FIFO：512 ワード/チャンネル
- L2 FIFO：8000 ワード
- 最大 6 台のボードの同期が可能
- 200 MB/s のスループットを持つ PCIe 1.1 x1
- 補助トリガー付き ADC 入力

1.2 用途

TDC8HQ は、単一ボードで最大 8 チャンネル、6 台のボードを同期させた場合は最大 48 チャンネルまでのすべての時間測定用途に使用できます。ストリーミング TDC の柔軟性を必要としない 4 チャンネル以下の用途向けには、当社ウェブサイト（www.cronologic.de）にてよりシンプルな製品を提供しています。

TDC8HQ は以下の用途に最適です：

- 分割型検出器を備えた飛行時間型質量分析計（TOF-MS）
- 自動テスト設備（ATE）
- コインシデンス測定
- 量子鍵配送（QKD）
- 時間相関単一光子計数法（TCSPC）
- 原子時計の同期
- 蛍光寿命イメージング顕微鏡（FLIM）

2 ハードウェア

2.1 PCIe ボードの取り付け

TDC8HQ ボードは、x1 以上のレーンを持つ任意の PCIe-CEM スロットに取り付けることができます。ボードを取り付ける際は、PC の電源がオフになっており、主電源コネクタが取り外されていることを確認してください。

2.2 TDC8HQ の入力およびコネクタ

図 2.1 はスロットブランケット上の入力位置を示しています。

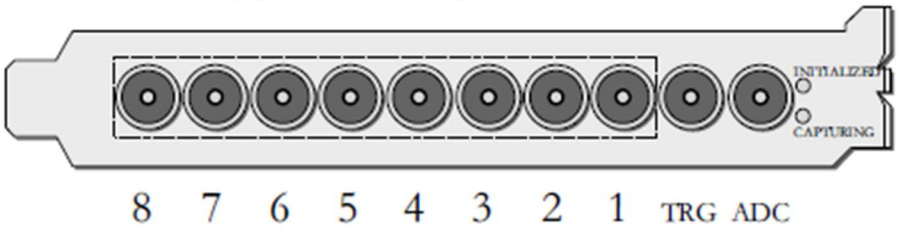


図 2.1 はスロットブラケット上の入力端子の位置を示しています。注意：TRG コネクタは ADC チャンネル専用のトリガーとして機能し、TDC チャンネル用ではありません。セクション 3.5 参照）

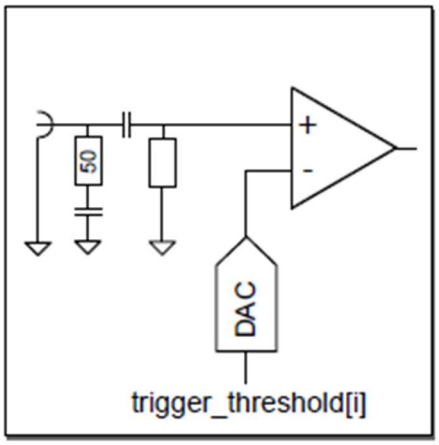


図 2.2 各入力チャンネルの入力回路。注意：TRG コネクタは ADC チャンネル専用のトリガーとして機能し、TDC チャンネル用ではありません。セクション 3.5 参照）

入力接続には LEMO-00 コネクタが使用されます。入力は AC 結合、50 Ωインピーダンスです。入力回路の概略図を図 2.2 に示します。各入力のデジタルしきい値は、多数のシングルエンド信号規格に適合するように調整できます。このしきい値は、正パルスまたは負パルスのいずれかに入力を設定するためにも使用できます。

これらのコネクタは出力としても使用できます。自動内部トリガーや外部機器の制御のための AC 結合出力パルスは、TiGer タイミングパターンジェネレータを使用して生成できます。TiGer の詳細についてはセクション 3.3 を参照してください。さらに、図 2.3 に示すように、TDC8HQ ボードの上面端付近に 3 つのボード間コネクタがあります。コネクタ

J25 は将来的な使用のために空けてあります。コネクタ J12 のピン配列を表 2.1 に、コネクタ J6 のピン配列を表 2.2 に示します。コネクタ J2 は同軸クロック入力です。セクション 2.4 で説明するように、複数のボードを組み合わせて使用する場合は 10 MHz クロックを受信する必要があります。

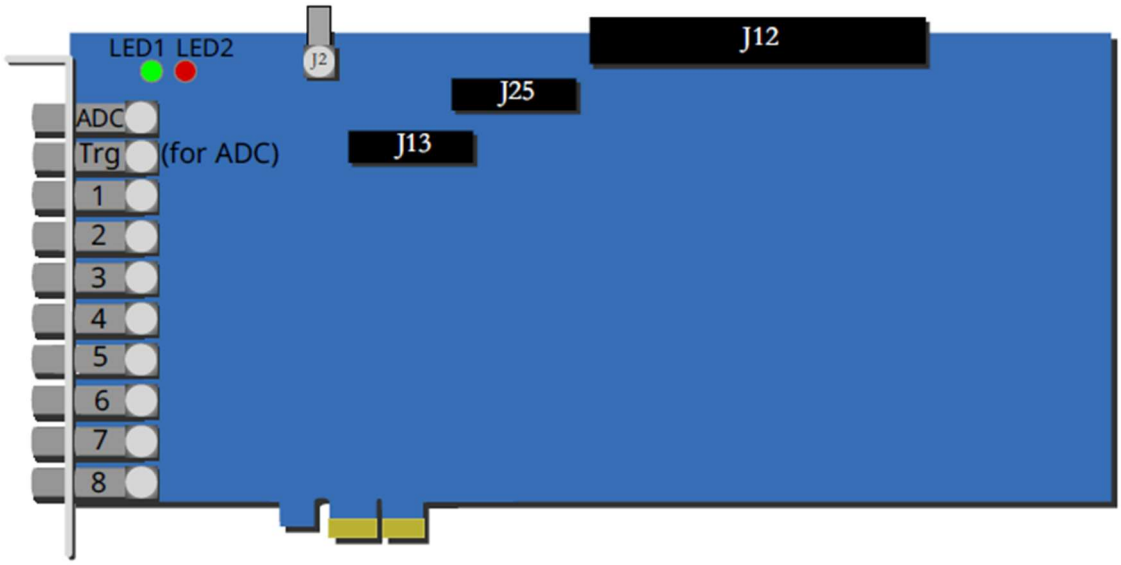


図 2.3 基板間コネクタを含む TDC8HQ ボード概略図

Pin	Name
1, 2	GND
3, 4	external CLK in N, external CLK in P
5, 6	GND
7, 8	reserved/NC
9, 10	GND
11, 12	reserved/NC
13, 14	GND
15, 16	reserved/NC
17, 18	GND
19, 20	reserved/NC
21, 22	GND
23, 24	reserved/NC
25, 26	GND
27, 28	reserved/NC
29, 30	GND
31, 32	reserved/NC
33, 34	GND

表 2.1 J12 ピン配列

Pin	Name
1	+3.3 V
2 - 9	reserved/NC
10	GND

表 2.2 J6 ピン配列

2.3 ステータス LED

ボード上には 4 つのステータス LED があります。2 つは PCB 上、もう 2 つは PCIe ブラケット上にあります。

- PCIe ブラケット上（図 2.1）：上の LED は、ボードの初期化完了後に緑色に点灯します。下の LED は、ボードのコンフィギュレーション中に短時間点灯し、カードがキャプチャを実行している間は点灯し続けます。
- PCB 上（図 2.2）：赤色の LED2 は、FPGA のコンフィギュレーション中に点灯し、完了後に消灯します。点灯したままの場合はコンフィギュレーションに失敗しています。緑色の LED1 は、ボードの初期化完了後に点灯します。

2.4 複数ボードの同期

8 つを超える TDC 入力が必要な場合、1 つのシステム内で最大 6 枚のボードを同期可能です。

4 章で説明する TDC8HQ API は、最大 6 枚のボードを自動的に管理し、すべてのボードからのヒットデータを時系列順に整列した単一のデータストリームとして提供します。各ボードのチャンネル A には、ボードインデックス×10 のチャンネル番号が割り当てられます。ボードインデックスは、0 から始まるシリアル番号の順にボードへ割り当てられます。

2.4.1 複数ボードの接続

各ボードは、コネクタ J2 で共通の 10 MHz クロック信号を受信する必要があります。このコネクタは PC ケース内部にあります。すべてのボードのコネクタ J12 は、両端にターミネータが付いたフラットリボンケーブルで接続する必要があります。ケーブルとターミネータは cronologic から入手可能です。図 2.4 は配線例です。

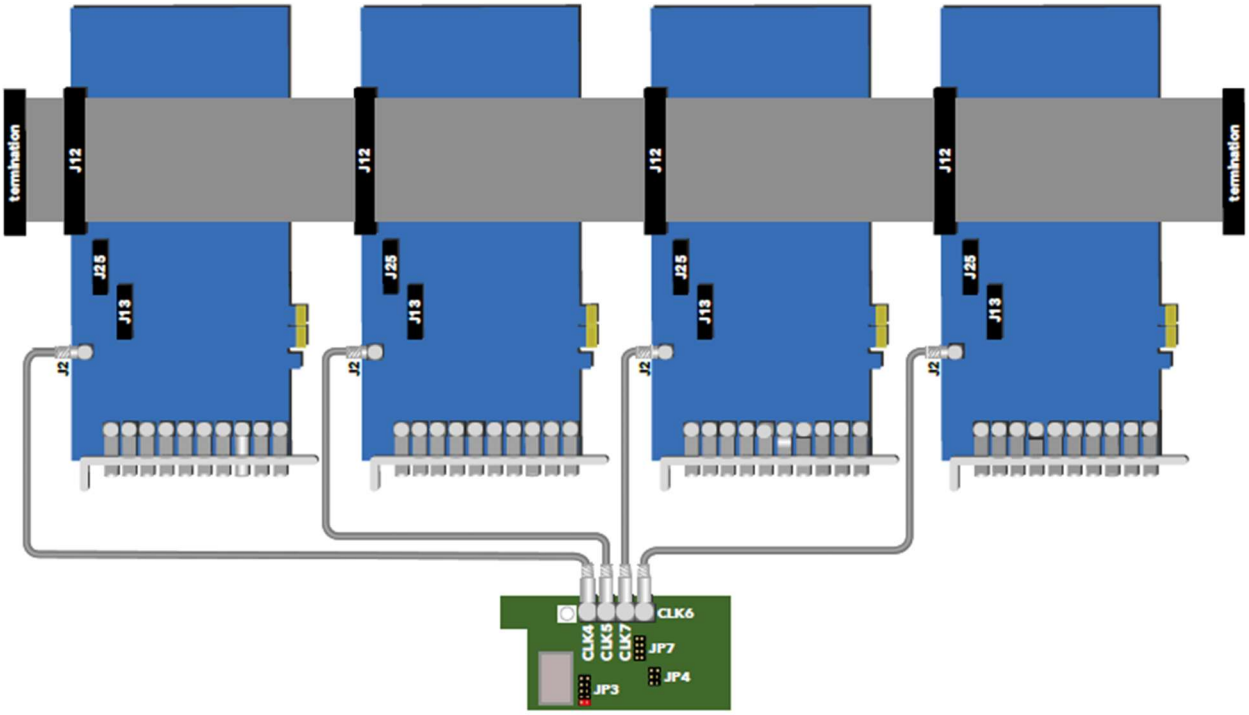


図 2.4 クロックボックスによる複数ボードの同期

2.4.2 ClockBox

最大 4 枚のボードで構成されるシステム向けに、cronologic は PC ケース内部で 4 つのクロック信号を簡便に供給できる ClockBox という製品を提供しています。TDC8HQ で使用する場合、クロック周波数を 10 MHz に設定するため、ClockBox のジャンパ JP3 を図 2.5 に示す通りに設定する必要があります。

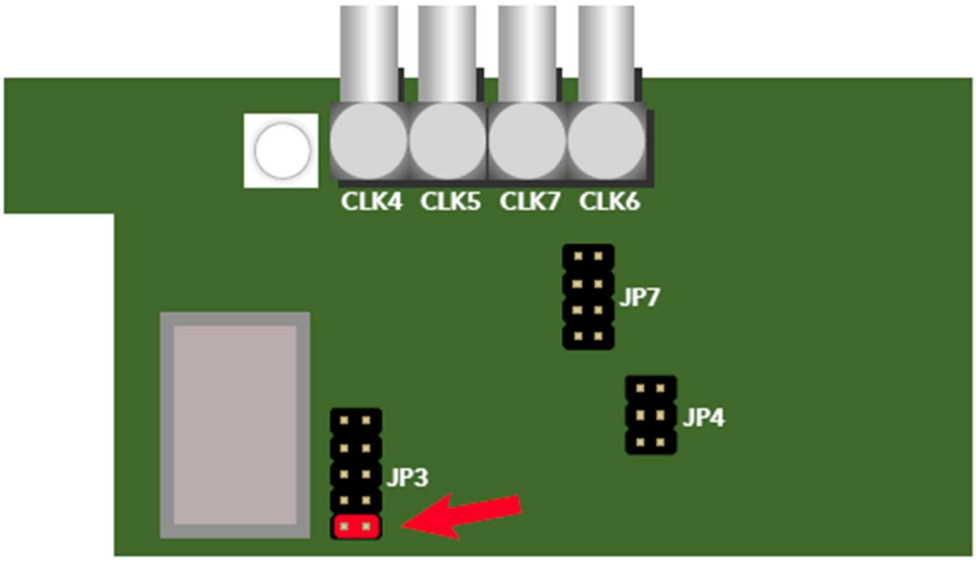


図 2.5 10 MHz 用の ClockBox ジャンパ設定

2.4.3 複数ボード用クレート

ほとんどの PC メインボードには、6 枚の TDC8HQ をサポートするのに十分な PCIe スロットがありません。弊社では、PCIe ケーブル延長技術を用いて、システムで利用可能なスロット数を拡張する「Ndigo Crate」という外付け筐体を提供しています。この拡張はホストシステムに対して完全に透過的です。追加のドライバーは不要です。弊社のウェブサイト（www.cronologic.de）の製品ページをご参照ください。

3 TDC8HQ の機能

TDC8HQ は、ストリーミング方式の時間-デジタル変換機です。入力 A～H における変化のタイムスタンプを無制限のストリームとして記録します。一般的なコモンスタートやコモンストップの動作をエミュレートできるフレキシブルなグルーピングモードが用意されています。詳細はセクション 3.1 をご参照ください。

各チャンネルにおいて、立ち上がりエッジと立ち下がりエッジのどちらを記録するかを個別に選択できます。同じチャンネルで両方のエッジを記録することはできません。タイムスタンプは、5000/(3×128) = 13.02083 ps のビンサイズの整数倍として記録されます。同じ入力チャンネルにおいて連続するヒットを確実に検出するには、ヒット間に少なくとも 5 ns の間隔が必要です。TDC8HQ は、約 48 MHits/s の読み出しレートにおいてデッドタイムなしでイベントを記録します。単一チャンネルにおける最大読み出しレートは約 12 MHits/s です。

3.1 グルーピングとイベント

一般的な用途では、1 つのスタートヒットに続いて多数のストップヒットが送られてきます。グルーピングが有効になっている場合、記録されたヒットはグループ（一部の用途では「イベント」と呼ばれる）ごとに管理されます。

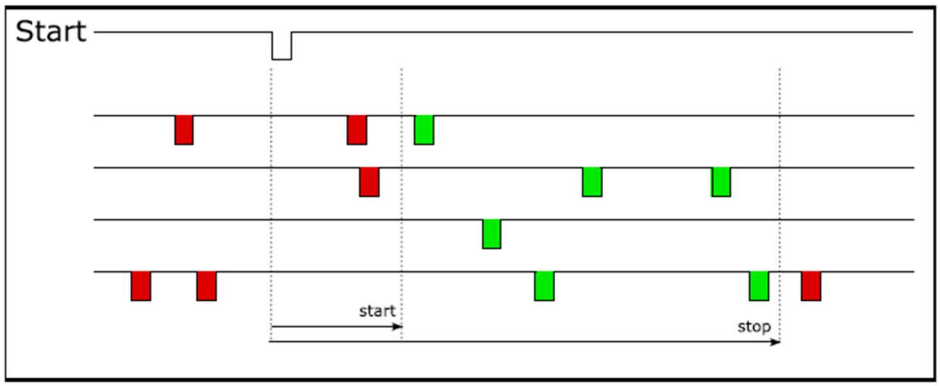


図 3.1 本文で説明されているように、取得されたヒットはグループに統合されます

図 3.1 に対応するタイミングチャートを示します。ユーザーはグループの範囲、すなわちストップチャンネル上のヒットが記録される時間枠を定義できます。そのタイムウィンドウの外側で発生したヒットは破棄されます。

グルーピングのスタート値およびストップ値には、負の値も設定可能です。これにより、コモンスタート、コモンストップ、あるいは両方向に広がるグループ用に TDC8HQ をセットアップすることができます。

これらの値は ps（ピコ秒）単位で設定します。

$$-2^{31} \le start \le stop \le 2^{31} - 1$$

シングルボード構成では、ゲーティングブロック（P28 参照）に対してもグルーピング機能と同様のパラメーターを設定することが推奨されます。これにより、グルーピングコード側で結局破棄されるデータの余計な読み出しを防ぐことができます。なお、グルーピングのパラメーターは ps 単位で指定するのに対し、ゲーティングブロックは 150 MHz クロックのサイクル数で設定することにご注意ください。ゲーティング機能の詳細については、セクション 3.3.4 をご参照ください。

グルーピングモードでは、`xHPTDC8_read_hits()` を呼び出すたびに、特定のトリガーイベントに対する相対的なタイムスタンプのグループが返されます。各グループの先頭には、チャンネル番号「255」を持つ追加のヒット情報が返さ

れます。このヒットにはグループの絶対時間が含まれています。残りの各ヒットの絶対時間は、この絶対時間に各ヒットの相対時間を加算することで取得できます。グルーピングモードを使用しない場合、この関数呼び出しは `xHPTDC8_start_capture()` からカウントアップされたすべてのタイムスタンプを絶対時間として返します。

3.2 オートトリガー・ファンクションジェネレータ

一部の用途では、周期的またはランダムな内部トリガーが必要となります。TDC8HQ のファンクションジェネレータはこの機能を提供します。

このトリガージェネレータにおける 2 つのトリガーパルス間の遅延は、固定値 M と、指数 N で与えられる範囲を持つ疑似乱数値の 2 つの要素の和となります。

周期は以下の通りです。

$$T = M + [1...2^N] - 1$$

1 クロックサイクルあたりの時間は 1/150 MHz=6.67 ns です。ここで、$6 \le M < 2^{32}$ かつ $0 \le N < 32$ です。
このトリガーは TiGer ユニット（セクション 3.3 参照）のソースとして使用できます。

3.3 タイミングジェネレータ（TiGer）

各デジタル LEMO-00 入力は、AC 結合のトリガー出力として使用できます。TiGer 機能はコネクタごとに独立して設定可能です。設定オプションの詳細については、セクション 4.5.5 を参照してください。

図 3.2 は、TiGer ブロックの接続方法を示しています。オートトリガーや ADC を含む任意の入力の論理和（OR）によってトリガーをかけることができます。各 TiGer は、対応する LEMO コネクタへ出力を駆動できます。これにより、そのコネクタが出力端子として機能するようになります。

TiGer 出力はコネクタに対して AC 結合されています。以下のいずれかのモードで動作させることができます。

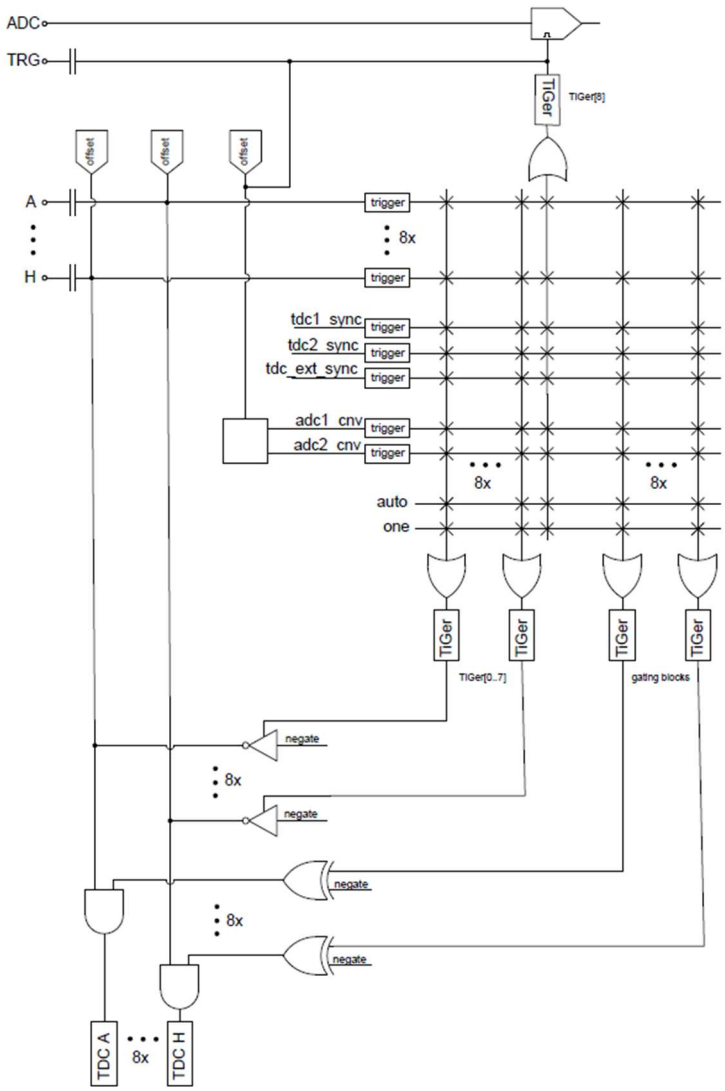
`XHPTDC8_TIGER_OFF`
: 信号はコネクタへ出力されません。

`XHPTDC8_TIGER_OUTPUT`
: このモードでは、コネクタが出力専用となります。パルスは振幅 2 V の単極です。接続された外部機器は出力として使用しているコネクタに信号を入力してはいけません。TDC8HQ および外部機器の両方を破損する恐れがあります。AC 結合による不必要なベースラインシフトを防ぐため、短いパルス幅での使用を推奨しますが、デバイス自体はデューティ比に対する制限を設けていません。このモードは、整数 N に対して 75/N MHz の周波数を持つクロック出力として使用できます。

`XHPTDC8_TIGER_BIDI`
: このモードでは、TiGer は振幅 1 V の単極パルスを生成します。コネクタは入力としても引き続き使用可能です。衝突の確率とベースラインへの影響を低く抑えるため、短いパルス幅を使用してください。

`XHPTDC8_TIGER_BIPOLAR`
: このモードでは、コネクタは振幅 1 V の双極パルスを生成します。コネクタは入力として使用可能です。このパルスはベースラインオフセットに影響を与えません。最小幅 2×6.67 ns の双極パルスを生成するには、TiGer を

stop = start に設定する必要があります。最大パルス幅は
`XHPTDC8_TIGER_MAX_BIPOLAR_PULSE_LENGTH = 15` です。



3.3.1 トリガーソース

TiGer およびゲーティングブロックのトリガーソースは、ビットマスクによって設定します。任意の数のトリガーソースをビットごとの論理和で組み合わせることができます。選択されたトリガー入力のいずれかがアクティブになると、ブロックにトリガーがかかります。

`XHPTDC8_TRIGGER_SOURCE_NONE`

トリガーソースを選択しない空のパターンです。

`XHPTDC8_TRIGGER_SOURCE_A ～ _H`

TDC の LEMO 入力端子です。

`XHPTDC8_TRIGGER_SOURCE_TDC1_SYNC`

`XHPTDC8_TRIGGER_SOURCE_TDC2_SYNC` と同じです。150/1024 MHz ≈ 146.5 kHz のクロック信号です。

XHPTDC8_TRIGGER_SOURCE_TDC_EXT_SYNC

125 kHz のクロック信号です。

XHPTDC8_TRIGGER_SOURCE_AUTO

オートトリガーブロックからの周期的またはランダムなトリガーパルスです。

XHPTDC8_TRIGGER_SOURCE_ADC1_CONV

`XHPTDC8_TRIGGER_SOURCE_ADC2_CONV` と同じです。TRG コネクタに ADC トリガーパルスが入ると、基板上の 2 つの ADC のうちいずれかが予測不能な不規則パターンでトリガーされます。TRG 入力をトリガーとして使用する場合は、トリガーソースに `ADC1_CNV` と `ADC2_CNV` の両方を含める必要があります。

XHPTDC8_TRIGGER_SOURCE_SOFTWARE

ドライバー関数 `xhptdc8_software_trigger()` の呼び出しにより、1 クロックサイクルの間有効化されます。

XHPTDC8_TRIGGER_SOURCE_ONE

毎クロックサイクルで常に有効となるトリガーです。

3.3.2 TiGer の例：200 kHz スタートパルスの生成

オートトリガーを使用して 200 kHz の周期を生成します。TiGer は、LEMO-00 出力 A 上にパルス幅 13.3 ns の 200 kHz 信号を生成します。

```
// generate an internal 200 kHz trigger
config.auto_trigger_period = 750;
config.auto_trigger_random_exponent = 0;

// setup TiGer
config.tiger_block[0].mode = XHPTDC8_TIGER_BIPOLAR;
config.tiger_block[0].start = 0;
config.tiger_block[0].stop = config.tiger_block[0].start + 1;
config.tiger_block[0].negate = 0;
config.tiger_block[0].sources = XHPTDC8_TRIGGER_SOURCE_AUTO;

// configure offset such that a 1V pulse can be detected by input A
config.trigger_threshold[0] = XHPTDC8_P_THRESHOLD_NIM;
```

3.3.3 TiGer の例：複数ソースからの遅延出力

チャンネル B から D のいずれかにおけるトリガーイベントを使用して、設定可能な遅延を持つバイポーラ出力パルスをチャンネル A 上に生成します。

```
config.tiger_block[0].mode = XHPTDC8_TIGER_BIPOLAR;
config.tiger_block[0].start = 20;
config.tiger_block[0].stop = config.tiger_block[0].start + 1;
config.tiger_block[0].negate = 0;
// an event on any of the channels B - D starts the TiGer
config.tiger_block[0].sources = XHPTDC8_TRIGGER_SOURCE_B|↩
   XHPTDC8_TRIGGER_SOURCE_C|XHPTDC8_TRIGGER_SOURCE_D;
```

3.3.4 TiGer による ADC のトリガー

ADC のトリガー入力（TRG, 図 2.1 参照）に接続された 9 番目の TiGer が存在します。ADC に関する追加情報についてはセクション 3.5 を参照してください。

再トリガーを有効にすると、ADC TiGer を使用して ADC データを周期的にサンプリングできます。周期は 300 ns または 45 TiGer クロック未満にしてはいけません。

ADC TiGer は、いずれかの TDC 入力に対する相対的なタイミングで電圧をサンプリングするためにも使用できます。この場合、パルスが連続して到来した場合でもサンプル周期の基準を満たせるよう、stop を少なくとも 45 に設定する必要があります。典型的な用途としては、スタート信号ごとに 1 回、低速な制御電圧をサンプリングすることなどが挙げられます。

3.4 ゲーティング

各 TDC チャンネルには、図 3.2 に示すようにゲートとして機能する 2 つ目の同等な TiGer ブロックが存在します。ドライバの設定構造体（セクション 4.5.6 参照）で、これらは `gating_block[channel]` としてアクセス可能です。

チャンネル内のヒットはゲーティングブロックが開いている間のみ記録されます。これは Gate Start、Gate Stop、negate、および retrigger によって設定されます。Gate Start、Gate Stop、retrigger はゲートのアクティブ／インアクティブのタイミングを操作し、negate はゲートがアクティブな間に開くか、インアクティブな間に開くかを制御します。negate が false の場合、出力がアクティブな間ゲートが開き、negate が true の場合は出力がアクティブな間ゲートは閉じます。

retrigger が有効な場合、ゲートがアクティブな間に 2 つ目のトリガーイベントが記録されるとゲートのタイマーがリセットされ、ゲートがアクティブである時間が延長されます。ゲートがアクティブになる前に 2 つ目のトリガーイベントが記録された場合、ゲートタイマーはゼロにリセットされます（すなわち、ゲートがアクティブになるまでに Gate Start の全時間が必要）。再トリガー無効の場合、Gate Stop 時間に達するまで二次的なトリガーイベントはすべて無視されます。

retrigger と negate のさまざまな組み合わせの概要を図 3.3 に示します。

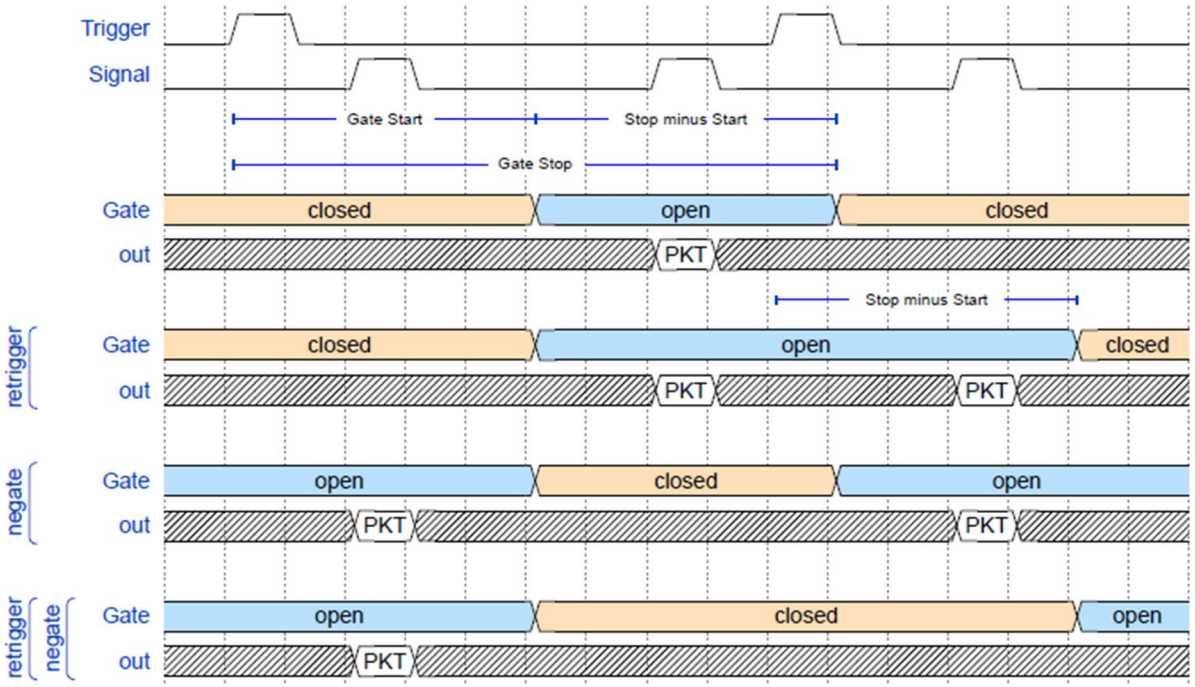


図 3.3 ゲーティングブロックの原理。パケット（PKT）はゲートが開いている間のみ記録されます。Gate Start、Gate Stop、retrigger、および negate は、ゲートが開くタイミングに影響を与えます。

ゲーティングは、トリガーが大量のノイズを生成するセットアップにおいて非常に有用な機能です。ゲーティングブロックを適切に設定することで、帯域幅とバッファの使用量を大幅に削減できます。ゲーティングは L0 バッファの前で実行されます。グルーピングは読み出し後にソフトウェア上で実行されます。

ボード内部の信号処理のため、ゲーティングロジックは即座には機能せず、約 6～7 クロックサイクルかかることに注意してください。これは、外部トリガーイベントが検出されてからゲーティングロジックが有効になるまでに、最大約 50 ns の一定のオフセットが存在することを意味します。（内部）オートトリガー機能を利用している場合、このオフセットは発生しません。

ゲーティング範囲を設定する際は、保守的な範囲を選択する（すなわち、最終的に必要となる範囲よりも広めに設定する）ことが推奨されます。これは、ゲーティングロジックの実装方法に起因して、ゲートの端において実際の信号エッジに対応しない偽タイムスタンプが生成される可能性があるためです。ゲートは最後にゲートがアクティブだったときからの入力チャンネルの状態を保持しており、ゲートが再度有効になると状態を更新します。この更新がエッジとして解釈されることがあり、その時点で入力信号の実際のエッジに必ずしも対応しないタイムスタンプが作成されます（図 3.4 も参照）。ただし、ゲートのエッジを跨がない信号のタイムスタンプは常に正確です。

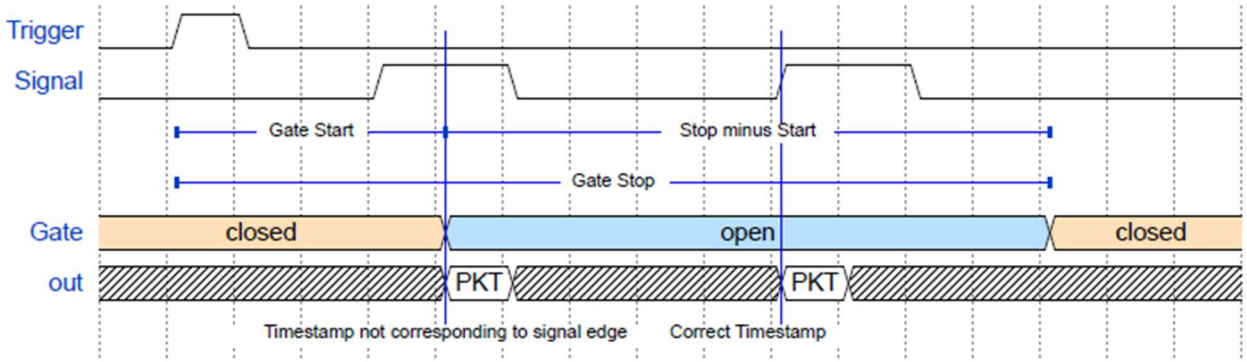


図 3.4 ゲートエッジにおいて、信号のエッジと一致しないタイムスタンプが生成される可能性のあるゲーティング構成の例。ゲートは、前回アクティブだった時点（この例では低信号レベル）の状態を維持し、再アクティブ化（高信号レベル）された時点でその状態を更新する。これがエッジとして記録される場合がある。

3.5 トリガー可能 ADC

TDC8HQ には、トリガー可能な ADC が搭載されています。ADC トリガーコネクタ（TRG、図 2.1 参照）で立ち上がりエッジが発生するたびに、ADC 入力コネクタの電圧がサンプリングされます。その結果は、タイムスタンプと ADC 値を持つパケットとして読み出しデータストリームに挿入されます。タイムスタンプの時間分解能は 833 ps です。

TRG は TiGer ブロックの出力にも接続されています。セクション 3.3.4 で説明されているように、これを使用して ADC を周期的、あるいは TDC 入力の 1 つに対して相対的なタイミングでトリガーすることができます。

ADC のトリガー間隔は、少なくとも 300 ns 以上離す必要があります。

ADC 入力パルスの幅は、13.2 ns より大きく、35 ns より小さくする必要があります。

読み出し中であっても常に ADC を利用可能にするため、2 つのインターリーブ（交互動作）ADC が搭載されています。これは出力データフォーマット、ならびに TiGer およびゲーティングのトリガーソースの両方でユーザーに公開されています。ADC トリガーをゲーティングまたは TiGer のトリガーとして使用する場合、両方のトリガーソースを同じ値に設定する必要があります。読み出し時、ユーザーが ADC データに対して高度なキャリブレーションを希望する場合を除き、2 つの ADC からのデータを区別する必要はありません。その場合に限り、2 つの ADC を個別に扱う必要があります。

4 ドライバープログラミング API

API は、C 言語のリンケージを持つ DLL です。

ドライバが提供する関数は `xHPTDC8_interface.h` 内で宣言されており、ソースコードにこのファイルをインクルードする必要があります。コンパイラに対し、ファイル `xhptdc8_driver.lib` または対応する 64 ビットバージョン `xhptdc8_driver_64.lib` をリンクするよう指定する必要があります。プログラムを実行する際、実際のドライバーコードを含む動的リンクライブラリ（DLL）は、実行可能ファイルと同じディレクトリに配置するか、PATH 環境変数に含まれるディレクトリに存在している必要があります。Linux の場合、静的ライブラリ `libxtdc4_driver.a` としてのみ提供されます。DLL 用ファイル名は `xTDC4_driver_64.dll` です。

これらすべてのファイルは、製品ウェブサイト（www.roentdek.com）からダウンロード可能なドライバーインストーラーに同梱されています。デフォルトでは、ファイルを `C:¥Program Files¥cronologic¥TDC8HQ¥driver` ディレクトリに配置します。コーディング例については RoentDek 社にお問い合わせください。

4.1 定数

`#define` **`XHPTDC8_MANAGER_DEVICES_MAX 6`**

デバイスマネージャーがサポートするボードの最大数。

`#define` **`XHPTDC8_TDC_CHANNEL_COUNT 8`**

TDC 入力チャンネル数。

`#define` **`XHPTDC8_GATE_COUNT 8`**

ゲーティングブロック数。各 TDC 入力に対して 1 つずつ存在します。

`#define` **`XHPTDC8_TIGER_COUNT 9`**

タイミングジェネレーター数。各 TDC 入力に対して 1 つずつと、ADC トリガー用に 1 つ存在します。

`#define` **`XHPTDC8_TRIGGER_COUNT 16`**

タイミングジェネレーターの潜在的なトリガーソース数。各 TDC 入力用に加えて、いくつかの特殊なソースが存在します。詳細についてはセクション 4.5.4 を参照してください。

`#define` **`XHPTDC8_OK 0`**

エラーが発生しなかった場合、API 関数によってエラーコードがこの値に設定されます。その他のエラーコードは `xHPTDC8_interface.h` 内で確認できます。

4.2 ドライバー情報

ボードが接続されていない状態であっても、以下の関数を使用してドライバのバージョンを確認できます。

`int` **`xhptdc8_get_driver_revision()`**

ドライバーのバージョンを返します（`xhptdc8_static_info.driver_revision` と同じフォーマット）。この関数は TDC8HQ ボードが存在していなくても動作します。

`const char*` **`xhptdc8_get_driver_revision_str()`**

SVN ビルドリビジョンを含むドライバーのバージョンを文字列として返します。

`int` **`xhptdc8_count_devices`**`(int` **`*error_code`**`, char` **`**error_message`**`)`

このドライバーがサポートする、システム内に存在するボードの数を返します。エラーコードおよびメッセージ用変数へのポインタを渡す必要があります。error_code が `#define XHPTDC8_OK = 0` と等しくない場合、エラーメッセージには問題の原因(例:crono カーネルが正常にインストールされていない)が含まれます。

4.3 初期化

データを読み出す前に、まずカードを初期化する必要があります。通常の手順としては、デフォルトの初期化パラメータを取得し、一部の値を変更します(例:インデックス指定による複数カードからの選択や、より大きなバッファサイズ)。

`int` **`xhptdc8_get_default_init_parameters(`**`xhptdc8_manager_init_parameters` **`*init)`**

標準パラメータを設定します。xhptdc8_init() 用のデフォルトパラメータセットを取得します。`xhptdc8_manager_init_parameters` 構造体を変更して `xhptdc8_init` へ渡す前には、必ずこの関数を使用して構造体を初期化しなければなりません。

`int` **`xhptdc8_init(`**`xhptdc8_manager_init_parameters` **`*params)`**

システム内の TDC8HQ ボードを開き、初期化します。

戻り値が `#define XHPTDC8_OK = 0` と等しくない場合、デバイスの初期化は失敗しています。パラメータ `params` は `xhptdc8_manager_init_parameters` 型の構造体へのポインタであり、`get_default_init_parameters()`によって完全に初期化されている必要があります。

`int` **`xhptdc8_close()`**

デバイスをクローズし、すべてのリソースを解放します。

4.3.1 構造体 xhptdc8_manager_init_parameters

`int` **`version = XHPTDC8_API_VERSION;`**

バージョン番号です。`XHPTDC8_API_VERSION` に設定されている必要があります。

`uint64_t` **`buffer_size = XHPTDC_DEFAULT_BUFFER_SIZE;`** `// 0x1000000 (16 MB)`

DMA バッファの最小サイズです。0 に設定した場合、デフォルトサイズの 16 MByte が使用されます。

`int` **`variant = 0;`**

`0` に設定します。異なるベース周波数など、将来のデバイスバリアントを有効化するために使用できます。

`int` **`device_type = CRONO_DEVICE_XHPTDC8;`**

cronologic の各種デバイス(`CRONO_DEVICE_*`)を表す定数です。

`xhptdc8_get_default_init_parameters()`によって初期化されます。この値は PCI Device ID と同一です。変更せずそのままにする必要があります。

```
#define CRONO_DEVICE_HPTDC        0x1
#define CRONO_DEVICE_NDIGO5G      0x2
#define CRONO_DEVICE_NDIGO250M    0x4
#define CRONO_DEVICE_xTDC4        0x6
#define CRONO_DEVICE_TIMETAGGER4  0x8
#define CRONO_DEVICE_XHPTDC8      0xC
#define CRONO_DEVICE_NDIGO6       0xD
```

`int` **`dma_read_delay = 250;`**

PCIe 経由でパケットが送信された後のライトポインタの更新遅延時間です。16 ns の倍数で指定します。ユーザーが変更すべきではありません。

`crono_bool_t` **`multiboard = false;`**

複数のデバイスを同期させる場合に true に設定します。また、クロックソースも外部クロックに設定されます。

`crono_bool_t` **`use_ext_clock = false;`**

`true` に設定すると J2 コネクタの外部 10 MHz 参照クロックを使用し、`false` に設定すると内部クロックソースを使用します。`multiboard` が **`true`** に設定されている場合、このパラメータは無視され、外部クロックが使用されます。

`crono_bool_t` **`ignore_calibration = false;`**

デバイスのキャリブレーションデータを使用するため、**`false`** のままにしてください。

4.4 ステータス情報

4.4.1 情報取得用関数

ドライバーは、ボードの種類、設定、パラメータ、および状態に関する詳細情報を取得するための関数を提供します。
情報は、その適用範囲やボードから情報を問い合わせる際の計算コストに応じて分類されています。
情報はボードごとに提供されます。パラメータの **`index`** で、どのボードに問い合わせるか選択します。

`int xhptdc8_get_device_type(int index)`

デバイスの種類を **`CRONO_DEVICE_XHPTDC8`** として返します。

`const char* xhptdc8_get_last_error_message(int index)`

最新のエラーメッセージを返します。index が負の値の場合、**`xhptdc8_manager`** からの最新のエラーメッセージが返されます。そうでない場合は、選択されたボードの最新のエラーメッセージが返されます。

`int xhptdc8_get_fast_info(int index, xhptdc8_fast_info *info)`

高速ダイナミック情報を返します。

数マイクロ秒以内に取得可能な動的情報を含む構造体を取得します。

`int xhptdc8_get_param_info(int index, xhptdc8_param_info *info)`

設定の変更情報を返します。

設定変更によって間接的に変化する情報を含む構造体を取得します。

`int xhptdc8_get_static_info(int index, xhptdc8_static_info *info)`

静的情報が含まれています。

実行中に変更されないボードに関する情報を含む構造体を取得します。

`int xhptdc8_get_pcie_info(int index, crono_pcie_info *pcie_info)`

PCIe 情報を読み出します。

修正可能エラーや修正不可能なエラーなど、PCIe の状態に関する情報を含む構造体を取得します。

`int xhptdc8_clear_pcie_errors(int index, int flags)`

PCIe エラーをクリアします。

PCIe のデバックにおいてのみ有用です。フラグは以下のいずれかになります。

```
#define CRONO_PCIE_CORRECTABLE_FLAG 1
#define CRONO_PCIE_UNCORRECTABLE_FLAG 2
```

int xhptdc8_get_temperature_info(int index, xhptdc8_temperature_info *info)

ボード上の複数のソースから温度測定値を取得します。

int xhptdc8_get_clock_info(int index, xhptdc8_clock_info *info)

クロック設定とステータス情報を取得します。

const char* xhptdc8_device_state_to _str(int state)

ステータス値を `xhptdc8_fast_info.state` から人が読める形式の文字列へ変換します。

4.4.2 Structure xhptdc8_static_info

この構造体は実行中に変更されないボードについての情報を含みます。`xhptdc8_get_static_info()`の関数で与えられます。

int size

構造体が締めるバイト数。

int version

構造体の定義が変更された際に増加するバージョン番号。インクリメントはドライバーのバージョン番号などに合わせて 1 より大きくなる場合があります。

int board_id

ボードの ID。システム内のすべての TDC8HQ ボードは、シリアル番号の順に 0 から始まる番号が割り当てられます。 ボードのチャンネル A は、チャンネル番号 $index \times 10$ を持ちます。

int driver_revision

エンコードされたドライバーのバージョン番号。 下位 3 バイトには 3 段階の階層構造を持つバージョン番号が含まれます（例：0x010103 はバージョン 1.1.3 をエンコード）。 バージョンは https://semver.org で定義されているセマンティック バージョニングのスキームに従います。最初の桁の変更は、通常、ユーザーアプリケーションの再コンパイルを必要とします。2 番目の桁の変更は、互換性を損なわない重要な改善または変更を示し、3 番目の桁は、API に影響を与えない軽微なバグ修正や同様の更新で増加します。

int driver_build_revision

cronologic の内部バージョニング システムに従ったドライバーのビルド番号。

int firmware_revision

FPGA 設定のリビジョン番号。

int board_revision

ボードのリビジョン番号。ボードのリビジョン番号はレジスタから読み取ることができます。ボードの回路図が変更されたときに変更される 4 ビットの数字です。これはボードに印刷されているリビジョン番号と一致する必要があります。

int board_configuration

ボードの回路構成を示します。

同一の基板であっても、複数の異なるバリエーションとして部品が実装される場合があります。これはレジスタから読み取ることができる 8 ビットのコードです。

int subversion_revision

FPGA 構成ソースコードの Subversion リビジョン ID。

int chip_id[2]

各 TDC チップの 16 ビットの工場出荷時 ID。

int board_serial

シリアル番号。

年と一連番号（連番）が 8.24 フォーマットで連結されています。この番号は、ボード上の銀色ステッカーに印刷されているものと同一です。

uint32_t flash_serial_high

uint32_t flash_serial_low

フラッシュ メモリ チップの 64 ビットの製造元シリアル番号。

crono_bool_t flash_valid

0 以外の場合、ドライバーがボード上のフラッシュ メモリから有効なキャリブレーション データを検出し、それを使用していることを示します。この値は TDC8HQ には適用されません。

char calibration_date[20]

カードがキャリブレーションされた日時の DIN EN ISO 8601 文字列（YYYY-MM-DD HH:MM）。

4.4.3 構造体 xhptdc8_param_info

この構造体には、`xhptdc8_get_param_info()` によって提供される設定の変更情報が含まれます。

int size

構造体が占めるバイト数。

int version

構造体の定義が変更された際に増加するバージョン番号。インクリメントはドライバーのバージョン番号などに合わせて 1 より大きくなる場合があります。

double binsize

測定された TDC データのビンサイズ（単位：ps）。

int channels

ボードの TDC チャンネル数。

8 に固定されています。

int channel_mask

有効化された各入力チャンネルのビット割り当て。

チャンネル n が有効な場合、ビット $0 \le n < 8$ が 1（セット）に設定されます。

int64_t total_buffer

DMA バッファーの総容量（バイト単位）。

4.4.4 構造体 xhptdc8_fast_info

int size

構造体が占めるバイト数。

int version

構造体の定義が変更された際に増加するバージョン番号。インクリメントはドライバーのバージョン番号などに合わせて 1 より大きくなる場合があります。

int fpga_rpm

FPGA ファンの回転数。ファンが存在しない場合は 0 を報告します。

int alerts

温度センサーおよびシステム モニターからのアラートビット。TDC の温度が 140 °C を超えるとビット 0 がセットされます。この場合、TDC はシャットダウンしているため、デバイスの再初期化が必要です。

int pcie_pwr_mgmt

常に 0。

int pcie_link_width

カードが使用する PCIe レーン数。TDC8HQ の場合は常に 10 となるはずです。

int pcie_max_payload

1 回の PCIe トランザクションにおける最大サイズ（バイト単位）。システム構成に依存します。

int state

TDC8HQ の現在のステータス。

```
const static int XHPTDC8_DEVICE_STATE_CREATED      0
const static int XHPTDC8_DEVICE_STATE_INITIALIZED  1
const static int XHPTDC8_DEVICE_STATE_CONFIGURED   2
const static int XHPTDC8_DEVICE_STATE_CAPTURING    3
const static int XHPTDC8_DEVICE_STATE_PAUSED       4
const static int XHPTDC8_DEVICE_STATE_CLOSED       5
```

4.4.5 構造体 crono_pcie_info

uint32_t pwr_mgmt

PCIe レーンの電力管理を制御・構成します。

uint32_t link_width

カードが使用する PCIe レーン数。

uint32_t max_payload

1 回の PCIe トランザクションにおける最大サイズ（バイト単位）。
システム構成に依存します。

uint32_t link_speed

PCIe カードのデータ転送速度。
システム構成に依存します。

uint32_t error_status_supported

このデバイスで PCIe エラー ステータスがサポートされている場合は 0 以外となります。

uint32_t correctable_error_status

PCIe 設定レジスタから直接取得された修正可能エラーのステータスフラグ。

PCIe の問題のデバッグに有用です。

```
#define CRONO_PCIE_RX_ERROR                  1 << 0
#define CRONO_PCIE_BAD_TLP                   1 << 6
#define CRONO_PCIE_BAD_DLLP                  1 << 7
#define CRONO_PCIE_REPLAY_NUM_ROLLOVER       1 << 8
#define CRONO_PCIE_REPLAY_TIMER_TIMEOUT      1 << 12
#define CRONO_PCIE_ADVISORY_NON_FATAL        1 << 13
#define CRONO_PCIE_CORRECTED_INTERNAL_ERROR  1 << 14
#define CRONO_PCIE_HEADER_LOG_OVERFLOW       1 << 15
```

uint32_t correctable_error_status

PCIe 設定レジスタから直接取得された修正不可エラーのステータスフラグ。

PCIe の問題のデバッグに有用です。

```
#define CRONO_PCIE_UNC_UNDEFINED                    1 << 0
#define CRONO_PCIE_UNC_DATA_LINK_PROTOCOL_ERROR     1 << 4
#define CRONO_PCIE_UNC_SURPRISE_DOWN_ERROR          1 << 5
#define CRONO_PCIE_UNC_POISONED_TLP                 1 << 12
#define CRONO_PCIE_UNC_FLOW_CONTROL_PROTOCOL_ERROR  1 << 13
#define CRONO_PCIE_UNC_COMPLETION_TIMEOUT           1 << 14
#define CRONO_PCIE_UNC_COMPLETER_ABORT              1 << 15
#define CRONO_PCIE_UNC_UNEXPECTED_COMPLETION        1 << 16
#define CRONO_PCIE_UNC_RECEIVER_OVERFLOW_ERROR      1 << 17
#define CRONO_PCIE_UNC_MALFORMED_TLP                1 << 18
#define CRONO_PCIE_UNC_ECRC_ERROR                   1 << 19
#define CRONO_PCIE_UNC_UNSUPPROTED_REQUEST_ERROR    1 << 20
```

4.4.6 Structure xhptdc8_temperature_info

この構造体は、ボード上のさまざまなチップの温度測定値を提供します。

int size

構造体が占めるバイト数。

int version

構造体の定義が変更された際に増加するバージョン番号。インクリメントはドライバーのバージョン番号などに合わせて 1 より大きくなる場合があります。

float tdc[2]

各 TDC チップの温度(単位:˚C)。

4.4.7 構造体 xhptdc8_clock_info

この構造体は、デバイスのクロックネットワークに関する情報を提供します。

int size

構造体が占めるバイト数。

int version

構造体の定義が変更された際に増加するバージョン番号。インクリメントはドライバーのバージョン番号などに合わせて 1 より大きくなる場合があります。

crono_bool_t cdce_locked

ジッタークリーニング PLL クロックシンセサイザーが同期状態を達成した場合に 1(セット)に設定されます。

int cdce_version

CDCE62005 クロックシンセサイザーからのバージョン情報。

crono_bool_t use_ext_clock

クロックシンセサイザーのソースは、通常、オンボードの 10 MHz 発振器です。初期化時には、代わりに J2 上の外部クロックを選択することもできます。複数のボードを同期させる場合、すべてのボードで共通の外部クロックを使用します。詳細についてはセクション 2.4 を参照してください。

crono_bool_t fpga_locked

FPGA データパス PLL が同期状態を達成した場合に 1(セット)に設定されます。

4.5 設定

システム内のすべての TDC8HQ ボードは、各ボードを設定するサブ構造体を包含する単一の設定構造体によって構成・設定されます。ユーザーはまず、ボード上の ROM から読み出されたデバイスの初期設定を含む構造体を取得し、必要に応じてユーザーアプリケーション向けに構造体を修正した上で、その結果を用いてデバイスを設定してください。

int xhptdc8_configure(xhptdc8_manager_configuration *config)

xhptdc8_manager のコンフィギュレーションを行います。

int xhptdc8_get_current_configuration(xhptdc8_manager_configuration *config)

現在の設定を取得します。指定された config ポインターへ現在の設定をコピーします。

int xhptdc8_get_default_configuration(xhptdc8_manager_configuration *config)

初期設定を取得します。指定された config ポインターへ初期設定をコピーします。

4.5.1 YAML 設定ファイル

TDC8HQ には、YAML 設定文字列から設定構造体を変更できる便利な関数を含む、コミュニティ維持のユーティリティ ライブラリが存在します。これにより、TDC のセットアップに必要なコードを大幅に短縮できます。
詳細については github.com/cronologic-de/xhptdc8_babel/tree/main/util を参照してください。

4.5.2 構造体 xhptdc8_manager_configuration

`int size`

構造体が占めるバイト数。

`int version`

構造体の定義が変更された際に増加するバージョン番号。インクリメントはドライバーのバージョン番号などに合わせて 1 より大きくなる場合があります。

`xhptdc8_device_configuration device_configs[XHPTDC8_DEVICES_MAX]`

セクション 4.5.3 で定義されている個々の TDC8HQ ボードの設定を含む構造体。有効な情報が含まれているエントリ数を照会するには、関数 xhptdc8_count_devices()を使用します。関数の詳細についてはセクション 4.2 を参照してください。

`xhptdc8_grouping_configuration grouping`

グルーピングのパラメーターを含む構造体。

構造体の定義についてはセクション 4.5.8、グルーピング詳細についてはセクション 3.1 を参照してください。

`int64_t *bin_to_ps(int64_t)`

将来の使用のために空けられています。

4.5.3 構造体 xhptdc8_device_configuration

これは設定情報を含む構造体です。ボードのさまざまな側面を設定するために複数のサブ構造体を使用します。

`int size`

構造体が占めるバイト数。

`int version`

構造体の定義が変更された際に増加するバージョン番号。インクリメントはドライバーのバージョン番号などに合わせて 1 より大きくなる場合があります。

`int auto_trigger_period = 300000000;`

`int auto_trigger_random_exponent = 0;`

周期的に、またはランダムにトリガーを生成します。2 つのパラメーター

M = auto_trigger_period

N = random_exponent

が存在し、トリガー間の間隔が T クロック サイクルとなります。

$T = 1 + M + [1...2^N]$

$0 \le M < 2^{32}$

$0 \le N < 32$

有効化やリセットはありません。オートトリガーは継続的に動作しています。このトリガーの使用方法は TiGer ブロックのソースフィールドで設定できます。

```
double trigger_threshold[XHPTDC8_TDC_CHANNEL_COUNT] = -0.35;
```

入力チャンネル A … H のしきい値電圧を設定します(図 4.1 を参照)。
サポートされている範囲は −1.32 ～ 1.18 V です。これは入力パルスの高さの 50%付近に設定する必要があります。さまざまなシグナリング規格の例は以下のように定義されています。

```
#define XHPTDC8_THRESHOLD_P_NIM          +0.35
#define XHPTDC8_THRESHOLD_P_CMOS         +1.18
#define XHPTDC8_THRESHOLD_P_LVCMOS_33    +1.18
#define XHPTDC8_THRESHOLD_P_LVCMOS_25    +1.18
#define XHPTDC8_THRESHOLD_P_LVCMOS_18    +0.90
#define XHPTDC8_THRESHOLD_P_TTL          +1.18
#define XHPTDC8_THRESHOLD_P_LVTTL_33     +1.18
#define XHPTDC8_THRESHOLD_P_LVTTL_25     +1.18
#define XHPTDC8_THRESHOLD_P_SSTL_3       +1.18
#define XHPTDC8_THRESHOLD_P_SSTL_2       +1.18
#define XHPTDC8_THRESHOLD_N_NIM          -0.35
#define XHPTDC8_THRESHOLD_N_CMOS         -1.32
#define XHPTDC8_THRESHOLD_N_LVCMOS_33    -1.32
#define XHPTDC8_THRESHOLD_N_LVCMOS_25    -1.25
#define XHPTDC8_THRESHOLD_N_LVCMOS_18    -0.90
#define XHPTDC8_THRESHOLD_N_TTL   -1.32
#define XHPTDC8_THRESHOLD_N_LVTTL_33     -1.32
#define XHPTDC8_THRESHOLD_N_LVTTL_25     -1.25
#define XHPTDC8_THRESHOLD_N_SSTL_3       -1.32
#define XHPTDC8_THRESHOLD_N_SSTL_2       -1.25
```

入力は AC 結合されています。そのため、パルス入力において絶対電圧は重要ではありません。入力回路を反転させるのは相対的なパルス振幅です。このパラメーターは、使用する入力規格に応じた相対的なスイッチング電圧に設定する必要があります。パルスが負極性の場合は負のスイッチングしきい値を設定し、逆の場合も同様とします。

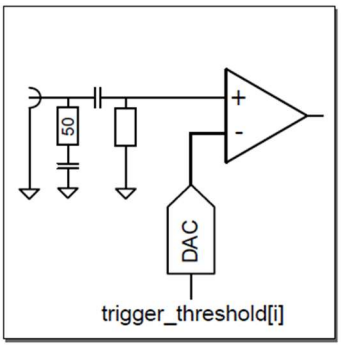


図 4.1 各入力チャンネルの入力回路

xhptdc8_trigger trigger[XHPTDC8_TRIGGER_COUNT]

外部トリガー ソースの極性設定(セクション 4.5.4 を参照)。これらは TiGer ブロックへの入力および時間測定ユニットへの入力として使用されます。

xhptdc8_tiger_block tiger_block[XHPTDC8_TIGER_COUNT]

タイミングジェネレーター(TiGer、セクション 4.5.5 を参照)の設定。

インデックス 0 から 7 はチャンネル A から H に対応し、インデックス 8 はチャンネル TRG に対応します。

xhptdc8_tiger_block gating_block[XHPTDC8_GATE_COUNT]

ゲーティングブロックの設定。

xhptdc8_channel channel[XHPTDC8_TDC_CHANNEL_COUNT]

TDC チャンネルの設定。

xhptdc8_adc_channel adc_channel

ADC チャンネルの設定。

crono_bool_t skip_alignment = false;

設定されている場合、キャプチャーの再開時に 2 つの TDC チップのアライメントを行いません。

int alignment_source = 1;

アライメントに使用される信号ソースを定義します。通常は変更せずにそのままにしておく必要があります。

```
#define XHPTDC8_ALIGN_TIGER          0
#define XHPTDC8_ALIGN_PIN            1
#define XHPTDC8_ALIGN_RESERVED       2
```

int alignment_off_state = 0;

使用していないときの TDC アライメントピンのステータスを選択します。

0 GND

1 VCCIO

2 high-Z

4.5.4 構造体 xhptdc8_trigger

各入力について、入力の立ち上がりエッジまたは立ち下がりエッジのどちらで TiGer およびゲーティングブロックへのトリガーイベントを発生させるかをこの構造体で決定します。

`crono_bool_t falling = true;`

`crono_bool_t rising = false;`

FPGA 内部でトリガーイベントを生成するエッジを選択します。TDC 自体は立ち上がりエッジまたは立ち下がりエッジのどちらか一方しか測定できませんが、ゲーティングブロックと TiGer はより柔軟です。TiGer またはゲーティングブロックで入力を使用する場合、いずれか、または両方のエッジの対応するフラグを設定します。

4.5.5 構造体 xhptdc8_tiger_block

詳細についてはセクション 3.3 を参照してください。

`int mode = 0;`

TiGer の希望する動作モードを有効にします。

`#define XHPTDC8_TIGER _OFF`	0	動作なし
`#define XHPTDC8_TIGER _OUTPUT`	1	振幅 2 V で出力が駆動されます。 入力を接続してはなりません
`#define XHPTDC8_TIGER _BIDI`	2	振幅 1 V で出力が駆動されます。 パルス レートは低くする必要があります。
`#define XHPTDC8_TIGER _BIPOLAR`	3	振幅 1 V の双方向パルスが出力駆動されます。 *start* = *stop* − 1

ゲーティング ブロックは内部でのみ使用され、ユーザーがアクセスできるパルスは生成されません。ゲーティング ブロックは 0 以外のすべての値を有効として解釈します。

`#define XHPTDC8_GATE _OFF`	0	ゲーティングなし、すべてのヒットがキャプチャーされます。
`#define XHPTDC8_GATE _ON`	1	ゲートが非アクティブの間はヒットがキャプチャーされません。

`crono_bool_t negate = false;`

出力の極性を反転します。デフォルトは `false` に設定されています。

ゲーティング ブロックの場合、`false` の値は *start* と *stop* の間の入力を有効にし、`true` の値はその間隔内の出力をブロックします。TiGer は反転されない限り、*start* から *stop* までハイパルスを生成します。

`crono_bool_t retrigger = false;`

再トリガー設定を有効にします。有効にすると、*stop* 時間に達するのを待つ間に入力信号が設定されるたびに、タイマーが *start* パラメーターの値にリセットされます。

`crono_bool_t extend = true;`

実装されていません。

`uint32_t start = 0;`

`uint32_t stop = 1000;`

20/3 ns = 6.666 ns の倍数単位。トリガー入力に対する、TiGer 出力が設定される時間。

ゲーティングブロックの場合、*start*/*stop* と外部入力信号が検出される時間との間に約 6～ サイクルの一定のオフセットが存在します（セクション 3.3.4 も参照）。

パラメーター *start* および *stop* は、以下の条件を満たす必要があります。

$$0 \le \texttt{start} \le \texttt{stop} \le 2^{16} - 1$$

再トリガーが有効な場合、stop 時間に達するのを待つ間に入力信号が設定されるたびに、タイマーが start パラメーターの値にリセットされます。

`int sources = 0x00000001;`

この TiGer ブロックをトリガーできるすべてのトリガーソースに対応するビットがセットされたビットマスク。デフォルトは **XHPTDC8_TRIGGER_SOURCE_A** です。

```
#define XHPTDC8_TRIGGER_SOURCE_NONE             0x00000000
#define XHPTDC8_TRIGGER_SOURCE_A                0x00000001
#define XHPTDC8_TRIGGER_SOURCE_B                0x00000002
#define XHPTDC8_TRIGGER_SOURCE_C                0x00000004
#define XHPTDC8_TRIGGER_SOURCE_D                0x00000008
#define XHPTDC8_TRIGGER_SOURCE_E                0x00000010
#define XHPTDC8_TRIGGER_SOURCE_F                0x00000020
#define XHPTDC8_TRIGGER_SOURCE_G                0x00000040
#define XHPTDC8_TRIGGER_SOURCE_H                0x00000080
#define XHPTDC8_TRIGGER_SOURCE_TDC1_SYNC        0x00000100
#define XHPTDC8_TRIGGER_SOURCE_TDC2_SYNC        0x00000200
#define XHPTDC8_TRIGGER_SOURCE_TDC_EXT_SYNC     0x00000400
#define XHPTDC8_TRIGGER_SOURCE_ADC1_CONV        0x00000800
#define XHPTDC8_TRIGGER_SOURCE_ADC2_CONV        0x00001000
#define XHPTDC8_TRIGGER_SOURCE_SOFTWARE         0x00002000
#define XHPTDC8_TRIGGER_SOURCE_AUTO             0x00004000
#define XHPTDC8_TRIGGER_SOURCE_ONE              0x00008000
```

4.5.6 構造体 xhptdc8_channel

TDC チャンネル設定が含まれます。

`crono_bool_t enable = false;`

TDC チャンネルを有効化します。

`crono_bool_t rising = false;`

TDC によって測定される信号のエッジを選択します。TiGer およびゲーティングブロックは、各入力で両方のエッジを同時に使用できる別個の設定を使用します（セクション 4.5.4 を参照）。

4.5.7 構造体 xhptdc8_adc_channel

この構造体は ADC 入力および対応するトリガー入力を設定します。セクション 3.5 を参照してください。

`crono_bool_t enable = false;`

ADC データの取得を有効化します。

crono_bool_t watchdog_readout = false;

出力データに定期的な ADC 測定値を含めます。ウォッチドッグ測定は、同時に発生する ADC トリガーを妨げません。

int watchdog_interval = 6144;

1 つのウォッチドッグ周期内における 150 MHz クロックサイクルの数。

$$100 \le \texttt{watchdog_interval} \le 7500$$

double trigger_threshold = -0.35;

TRG 入力のしきい値電圧。セクション 4.5.3 の channel.trigger_threshold の説明を参照してください。

4.5.8 構造体 xhptdc8_grouping_configuration

この構造体はグルーピング機能の動作を設定します（セクション 3.1 を参照）。

この構造体では、TDC のビンサイズに関係なく、間隔は常にピコ秒単位で指定されます。

crono_bool_t enabled = false;

グルーピングを有効化します。

int trigger_channel = 0;

グループの生成をトリガーするために使用されるチャンネル番号。

uint64_t trigger_channel_bitmask = 0ull;

追加のトリガーチャンネルを定義するために使用します。`trigger_channel` との OR 論理和となります。

int zero_channel = -1;

オプションで、グループ内の相対タイムスタンプを計算するために別のチャンネルを使用できます。このパラメーターを-1 に設定することでデフォルトで無効化されます。

int64_t zero_channel_offset = 0;

このピコ秒単位のオフセットがグループ内の相対タイムスタンプに追加されます。

int64_t range_start = -1500;

`trigger_channel` に対するグループ範囲の開始位置。

int64_t range_stop = 1500;

`trigger_channel` に対するグループ範囲の終了位置。

`range_start` から `range_stop` までの区間に含まれる値がグループに含まれます。コモンストップ動作を作成するために、一方または両方の値を負にすることができます。

$$-2^{63} \le \texttt{range_start} \le \texttt{range_stop} < 2^{63}$$

int64_t trigger_deadtime = 0;

トリガーが検出された後、この間隔の間、追加のトリガーは抑制されます。負の値にしてはなりません。

uint64_t window_hit_channels = 0ull;

チャンネルのビットマスクです。`window_start` および `window_stop` のウィンドウ内に 1 つ以上のヒットが存在する場合にのみグループが生成されます。使用方法は `trigger_channel_bitmask` と同等です。

int64_t window_start = 0;

int64_t window_stop = grouping.window_start + 1;

$$-2^{63} \le \texttt{window_start} \le \texttt{window_stop} < 2^{63}$$

int veto_mode = 0;

veto_start および veto_stop によって定義されるウィンドウを使用してヒットの抑制ができます。この機能はゲーティングブロックに非常に似ていますが、ソフトウェア上で定義されます。ゲーティングブロックが各ボード内で利用可能な情報に対してローカルにしか機能しないのに対し、veto(ベトー)機能はすべてのボードにわたってグローバルに適用されます。この機能は FIFO の使用率や PCIe 帯域幅の使用率を削減するために使用することはできません。ベトーウィンドウの内側または外側のデータのいずれかを抑制できます。

```
#define XHPTDC8_GROUPING_VETO_OFF          0
#define XHPTDC8_GROUPING_VETO_INSIDE       1
#define XHPTDC8_GROUPING_VETO_OUTSIDE      2
```

int64_t veto_start = 0;

int64_t veto_stop = 0;

$$-2^{63} \le \texttt{veto_start} \le \texttt{veto_stop} < 2^{63}$$

uint64_t veto_active_channels = 0xffffffffffffffffull;

ベトーが有効な場合、チャンネル ビットマスクによって定義されたチャンネルに対してベトーフィルタリングがアクティブになります。デフォルトでは、すべてのチャンネルに対してフィルタリングがアクティブになります。

crono_bool_t veto_relative_zero = false;

設定されている場合、ベトーウィンドウはゼロチャンネルに対して相対的に定義されます。設定されていない場合、ウィンドウはトリガーに対して相対的に定義されます。

crono_bool_t ignore_empty_events = false;

トリガー信号のみを含んでいたグループを破棄します。

crono_bool_t overlap = false;

サポートされていません。false のままにする必要があります。

4.6 ユーザーデータストレージ

各ボードには 64 kByte のフラッシュメモリが備わっており、ユーザーは任意の形式のデータを保存するために利用できます。典型的なユースケースとしては、TDC8HQ または本デバイスが接続されている検出器のキャリブレーションデータなどが挙げられます。また、TDC8HQ を組み込んで構築された機器のシリアル番号などもここに保存できます。書き込み処理を行うと、常にメモリブロック全体が消去されます。

#define XHPTDC8_USER_FLASH_SIZE 0x10000

フラッシュメモリのサイズ(バイト単位)。

int xhptdc8_read_user_flash(int index, uint8_t* flash_data, uint32_t size)

int xhptdc8_write_user_flash(int index, uint8_t* flash_data, uint32_t size)

index で識別されるボードのユーザーフラッシュからの読み出し、またはユーザーフラッシュへの書き込みを行います。flash_data はユーザーによって割り当てられたバッファーを指している必要があります。size にはそのバッファーのサイズ(バイト単位)を指定する必要があります。常にバッファー全体が上書きされることを明確にするため、XHPTDC8_FLASH_SIZE で与えられるフラッシュメモリのサイズ分のバッファーを常に割り当てることを推奨します。

5 実行時制御(ランタイム コントロール)

5.1 ドライバーのステータス制御

デバイスの設定が完了すると、以下の関数を使用してデバイスの動作を制御できます。これらの関数はすべてわずかなオーバーヘッドで迅速に処理を返しますが、スレッドセーフであることは保証されません。

`int xhptdc8_start_capture()`
データ取得を開始します。

`int xhptdc8_pause_capture()`
開始されたデータ取得を一時停止します。一 *pause* および *continue* は、*start* および *stop* よりもオーバーヘッドが小さくなりますが、設定の変更を行うことはできません。

`int xhptdc8_continue_capture()`
`xhptdc8_pause_capture()` を呼び出した後に、データ取得を再開するために呼び出します。pause および continue は、start や stop よりもオーバーヘッドが小さくなりますが、設定の変更を行うことはできません。

`int xhptdc8_stop_capture()`
データ取得を停止します。

`int xhptdc8_start_tiger(int index)`
`int xhptdc8_stop_tiger(int index)`
個々のボードのタイミング ジェネレーターを開始および停止します。これはデータ取得のステータスとは独立して行うことができます。

`int xhptdc8_software_trigger(int index)`
ソフトウェア トリガーを 1 クロック サイクル分設定(生成)します。これは、トリガー ソース `XHPTDC8_TRIGGER_SOURCE_SOFTWARE` として TiGer およびゲーティングブロック向けに設定可能です。

5.2 データ読み出し

`int xhptdc8_read_hits(TDCHit *hit_buf, size_t size)`
ユーザーが用意したバッファーへ多数のヒットを読み出します。読み出されたヒット数を返します。
グルーピングが有効な場合、単一のグループが読み出されます。グループがバッファーに対して大きすぎる場合、そのグループの残りのヒットは破棄されます。
グルーピングが無効な場合、バッファーのサイズに達するまで利用可能なすべてのデータが読み出されます。
このメソッドは常に即座に復帰(返答)します。ヒットが読み出されなかった場合は、すぐに再試行するのではなく、sleep() を呼び出すか CPU を他のプロセスに譲渡(yield)することが推奨されます。
size にはバッファーに格納できる要素数を必ず設定してください。
この関数はスレッド セーフではありません。読み出したデータを複数のスレッドで処理したい場合は、スレッドごとにデータを別々のバッファーへコピーする必要があります。

`int xhptdc8_get_current_timestamp(int index, int64_t *timestamp)`
選択された `TDC8HQ` の現在の内部タイムスタンプ カウンターの値をピコ秒単位で返します。

6 出力データフォーマット

6.1 メモリ管理

ホストバッファーは、ユーザーからの肯定応答(アック)があるまで、TDC8HQ によって記録されたデータが保持されるホストシステム上のメモリです。

ホストバッファーは DMA(Direct Memory Access)ドライバーによって管理されます。DMA ドライバーは、十分な空きメモリが存在する場合にのみホストバッファーへ書き込みを行います。つまり、承認されていない古いパケットが新しいパケットによって上書きされることはありません。
ホストバッファーが満杯になるとデータ損失の可能性があります。その場合、crono_packet::flags の CRONO_PACKET_FLAG_HOST_BUFFER_FULL ビットがセットされます。これを防止するには、解析ソフトウェアによって十分に速くパケットを肯定応答する必要があります。ホストバッファー満杯によるデータ損失が発生した場合は、crono_packet::flags の CRONO_PACKET_FLAG_TRIGGER_MISSED ビットがセットされます。

6.1.1 パケットの肯定応答

ホスト バッファー内のパケットは、肯定応答された場合にのみ上書きされます。これは、ndigo_acknowledge() を呼び出すことで手動で行うか、または ndigo_read() の呼び出し時に ndigo_read_in 構造体の acknowledge_last_read を true に設定することでドライバーにより自動的に行われます(セクション 5 を参照)。
あるパケットを肯定応答すると、それ以前のすべてのパケットも同時に肯定応答されます。
パケットを肯定応答すると即座にそのパケットは無効化され、そのコンテンツへアクセスを試みることは即座に危険な状態となることに注意してください。

6.1.2 TDC8HQ 内部のメモリ レイアウト

TDC8HQ は内部 FIFO(First-In, First-Out)メモリを使用します。これらの FIFO のうち「DMA FIFO」と呼ばれるものには、ホストシステムへ送信する準備ができたパケットがバッファリングされます。DMA FIFO がいかなる時点で満杯になった場合でも、影響を受けたパケットの crono_packet::flags の CRONO_PACKET_FLAG_DMA_FIFO_FULL ビットがセットされます。これは必ずしもデータが損失したことを意味するわけではありません。CRONO_PACKET_FLAG_TRIGGER_MISSED ビットがセットされている場合にのみ、データが損失したことを示します。
測定された各エッジに対して、TDC8HQ はピコ秒単位の 64 ビットタイムスタンプと追加情報を含む 3 つのフィールドで構成される 12 バイトのデータ構造体 TDCHit を生成します。

6.2 構造体 TDCHit

int64_t time
ヒットのタイムスタンプ(ピコ秒単位)。
グルーピング無効の場合、タイムスタンプは xhptdc8_start_capture() の呼び出しから継続的にカウントアップします。
グルーピング有効な場合、タイムスタンプはグループのトリガーや個別設定されたゼロ参照に対して相対的となります。グループの最初の TDCHit はチャンネル番号 255 を持ち、グループの絶対時間を提供します。各ヒットの絶対時間は、この値を各相対タイムスタンプに加算することで取得できます。

uint8_t channel

システム内の最初のボードにおいて、これは TDC チャンネル A～H に対しては 0～7、ADC データに対しては 8～9 となります。チャンネル 8 および 9 からのデータは、通常、同一チャンネルからのデータとして扱われる必要があります。その他のボードでは、チャンネル番号は board_id ×10 加算されます。グルーピングモードでは、各グループの最初のヒットはチャンネル番号 255 を持ち、グループの絶対時間を含みます。

uint8_t type

記録されたヒットのタイプに関する追加情報（セクション 6.2.1 を参照）。

uint16_t bin

ADC ヒットの場合、これにはサンプリングされた電圧が含まれます。TDC ヒットの場合、内容は未定義です。

6.2.1 TDCHit タイプ

TDC 測定値のタイプ情報

ヒットがチャンネル A～H での TDC 測定値である場合、TDCHit 構造体の type フィールドに対して以下のフラグが定義されています。

#define XHPTDC8_TDCHIT_TYPE_RISING 0x01

立ち上がりエッジ

#define XHPTDC8_TDCHIT_TYPE_ERROR 0x02

あらゆるタイプのエラー

#define XHPTDC8_TDCHIT_TYPE_ERROR_TIMESTAMP_LOST 0x04

L1 FIFO のオーバーフローにより消失したヒット

#define XHPTDC8_TDCHIT_TYPE_ERROR_ROLLOVER_LOST 0x08

内部エラーによる無効なタイムスタンプ

#define XHPTDC8_TDCHIT_TYPE_ERROR_PACKETS_LOST 0x10

DMA パケットの損失により消失したヒット

#define XHPTDC8_TDCHIT_TYPE_ERROR_SHORTENED 0x20

短縮された DMA パケットにより消失したヒット

#define XHPTDC8_TDCHIT_TYPE_ERROR_DMA_FIFO_FULL 0x40

L2 FIFO のオーバーフローにより消失したヒット

#define XHPTDC8_TDCHIT_TYPE_ERROR_HOST_BUFFER_FULL 0x80

ホスト バッファーのオーバーフローにより消失したヒット

ヒットが消失した場合、同じボードから次回読み出されるヒットにおいてエラーフラグがセットされます。

ADC 測定値のタイプ情報

ヒットがチャンネル 8 または 9 での ADC 測定値である場合、TDCHit 構造体の type フィールドに対して以下のフラグが定義されています。

#define XHPTDC8_TDCHIT_TYPE_ADC_INTERNAL 0x01

内部ストローブによってトリガーされた ADC 測定

#define XHPTDC8_TDCHIT_TYPE_ADC_ERROR 0x02

あらゆるタイプのエラー

#define XHPTDC8_TDCHIT_TYPE_ADC_ERROR_INVALID_TRIGGER 0x08

TRG 入力がタイミング要件に違反しました。データが破損している可能性があります

#define XHPTDC8_TDCHIT_TYPE_ADC_ERROR_DATA_LOST 0x10

いずれかのバッファーのオーバーフローにより ADC 測定値が消失しました

ヒットが消失した場合、同じボードから次回読み出されるヒットにおいてエラーフラグがセットされます。

7 テクニカルデータ

各ボードは最小値および最大値の列に記載された値に対して検査・テストが行われます。標準値は、最初に製造された 10 台のボードの平均値、または設計によって設定された値です。

7.1 パフォーマンス

7.1.1 TDC 測定特性

Symbol	Parameter	Min	Typical	Max	Units
INL	Integral nonlinearity			1	bins
DNL	Differential nonlinearity			0.5	bins
t_{Bin}	Binsize		5000/384 $13.02083\overline{3}$		ps ps
t_{DPfull}	Interval between edges	5			ns
$f_{Readout}$	Readout rate			48	MHits/s

7.1.2 タイムベース

Symbol	Parameter	Min	Typical	Max	Units
ΔT	Stability in temperature range −20 °C to 70 °C[1]			10	ppb
F	Initial calibration		<300	500	ppb
$\Delta F/F_1$	Aging first year			100	ppb
$\Delta F/F_{20}$	All inclusive aging 20 years			1000	ppb
	Warm-up[2]			3	min.

[1] −40 °C から+85 °C 間：1 時間の連続動作後に安定した周波数を基準とします。

[2] @+25 ˚C。F の±100 ppb 以内（ただし F は 1 時間の連続動作後に達した安定周波数とします）。

7.1.3 ADC

Symbol	Parameter	Min	Typical	Max	Units
t_{ADC}	Interval between ADC triggers	300			ns
f_{-3dB}	-3dB bandwidth		200		MHz
V_{LSB}	Voltage resolution		76		μV

7.2 電気的特性

7.2.1 電源

Symbol	Parameter	Min	Typical	Max	Units
P_{total}	Total power consumption			20	W
$VCC_{3.3}$	PCIe 3.3 V rail power supply voltage	3.1	3.3	3.5	V
$I_{3.3}$	PCIe 3.3 V rail input current			600	mA
VCC_{12}	PCIe 12 V rail power supply voltage	11.1	12.0	12.9	V
I_{12}	PCIe 12 V rail input current			1500	mA
VCC_{aux}	PCIe 3.3 V_{Aux} rail power supply voltage		3.3		V
I_{aux}	PCIe 3.3 V_{Aux} rail input current		0		mA

7.2.2 TDC 入力

TDC8HQ の入力は 50Ω終端を備えたシングルエンドの AC 結合です。

Symbol	Parameter	Min	Typical	Max	Units
V_{Base}	Input Baseline	0		5	V
$V_{Threshold}$	Trigger Level	V_{Base} – 1.32		V_{Base} + 1.18	V
t_{Pulse}	Pulse Length	2	5	200	ns
t_{Rise}	Pulse Edge 20% to 80%			10	ns
t_{Fall}	Pulse Edge 80% to 20%			10	ns
Z_P	Input Impedance		50		Ω
I_{Term}	Termination Current	–50	–20	50	mA

すべての入力は AC 結合されています。入力には非常に高い入力帯域幅要件があるため、これらの信号に対する過電圧保護回路は備えられていません。スロットカバーの電圧に対して 5 V を超える電圧、または−5 V を下回る電圧が入力に加わると、ボードに永久的な損傷を与える可能性があります。パルス幅 t_{Pulse} と平均パルス間隔の比率（連続信号の場合はデューティサイクルと呼ばれる）によって、入力ベースライン V_{Base} が影響を受けることに注意してください。すべてのデジタル入力は、TiGer から AC 結合パルスを出力できます。パルスによって損傷する可能性がある感度の高い機器が接続されている場合は、TiGer 出力を有効にしないよう特別の注意を払う必要があります。セクション 3.3 を参照してください。

7.2.3 ADC 入力

ADC 入力は 400 mV の差動終端電圧に DC 結合されています。これは、ADC が実際に検出する電圧が、入力を駆動するソースの出力インピーダンスに依存することを意味します。

Symbol	Parameter	Min	Typical	Max	Units
V_{in}	Input voltage	−2.0		2.5	V
V_{term}	Termination voltage		0.4		V
Z_{in}	Input Impedance		50		Ω

7.2.4 クロック入力 J2

クロック入力 J2 はシングルエンドで AC 結合されています。

Symbol	Parameter	Min	Typical	Max	Units
V_{PP}	Peak to Peak voltage	1		3.3	V
V_{cm}	Common mode voltage voltage	−3		3	V
V_{tck}	Clock termination voltage		0		V
Z_{in}	Input Impedance		50		Ω
D_{J2}	Duty cycle	45	50	55	%
f_{J2}	Frequency		10		MHz

7.3 DIN EN 61010-1 に基づく必要情報

7.3.1 製造元

TDC8HQ の製品情報:

RoentDek Handels GmbH & Co. KG
Im Vogelshaag 8
65779 Kelkheim
Germany HRB 3300-125 016 24973 (Amtsgericht Koenigstein 登記)
VAT-ID: DE113875060
PCI ベンダー ID: 0x1A13

7.3.2 使用目的およびシステムの組み込み

本デバイスは cronologic から納品されたそのままの状態では使用できません。エンドユーザーの用途を満たすには、専用ソフトウェアの開発が必要です。本デバイスは、エンドユーザーに配布される測定システムに組み込むためのシステムインテグレーター向けに提供されています。これらのシステムは通常、TDC8HQ、メインボード、筐体、アプリケーションソフトウェア、およびシステムを特定の検出器に接続するための追加の電子機器で構成されます。また、検出器と一体化される場合もあります。

TDC8HQ は、適切にシールドされた筐体に収容された PCIe 準拠のメインボード上で運用される場合、DIN EN 61326-1 に適合するように設計されています。規格に準拠した閉じた筐体内で運用される場合、本デバイスは EN 61010-1 で定義されているような危険を引き起こすことはありません。

電磁放射、ノイズ耐性、および安全性は、筐体の品質に大きく依存します。組み立てられたシステムが使用される国の適用規格、特にユーザーの安全性および電磁障害に関する規格、に適合していることを確認するのは、システムインテグレーターの責任です。

ボードを取り扱う際は、回路を静電気放電(ESD)から保護するための適切な対策を講じる必要があります。ボードを取り付ける前に、システムに供給されているすべての電源をオフにする必要があります。

7.3.3 保管に関する環境条件

本ボードは、動作していない間、以下の条件で保管する必要があります。

Symbol	Parameter	Min	Typical	Max	Units
T_{store}	ambient temperature	−30		60	°C
RH_{store}	relative humidity at 31°C noncondensing	10		70	%

7.3.4 動作に関する環境条件

本ボードは、以下の条件において動作するように設計されています。

Symbol	Parameter	Min	Typical	Max	Units
T_{oper}	ambient temperature	5		40	°C
RH_{oper}	relative humidity at 31°C	20		75	%

警告：該当チャンネルの TiGer が出力として設定されている間は、DC 結合された入力をそのチャンネルに絶対に接続しないでください（セクション 3.3 参照）。これを行うと、TDC8HQ および外部ハードウェアに永久的な損傷を与える可能性があります。

7.3.5 冷却

基本構成の TDC8HQ は空冷冷却を採用しており、一定量の風量を必要とします。ケースの設計によりボードへ十分なエアフローを提供できない場合は、Zalman ZM-SC100 のようなスロットクーラーをボードの隣に配置できます。また、ボードのオプションとしてアクティブ冷却も用意されています。

7.3.6 リサイクル

RoentDek は電子システムの製造元として、ID DE 48573152 にて「Stiftung Elektro-Altgeräte Register」に登録されています。

TDC8HQ はカテゴリー6「一般家庭以外での専用使用を目的とした小型情報機器および電気通信機器」に分類されます。2018 年以前に販売されたデバイスはカテゴリー9「産業専用の監視および制御機器」に分類されます。TDC8HQ の最終所有者は、これをリサイクルするか、ドイツ電気・電子機器法（ElektroG）の第 11 条および第 12 条に従ってボードを処理するか、または 40 ページに記載されている製造元の住所へ返送する必要があります。

東京インスツルメンツ TOKYO INSTRUMENTS
グローバルにネットワークを広げ、最先端の科学をお客様に提供
本　社：〒134-0088 東京都江戸川区西葛西6-18-14 T. ビル　Tel. 03-3686-4711
営業所：〒532-0003 大阪府大阪市淀川区宮原4-1-46 新大阪北ビル　Tel. 06-6393-7411
URL：https://www.tokyoinst.co.jp　Mail：sales@tokyoinst.co.jp
Group Company
UNISOKU 超高真空・極低温走査型プローブ顕微鏡 高速分光測定装置、クライオスタット
LOTIS TII Nd:YAGレーザー、Ti:Sレーザー OPOレーザー
● 本カタログに記載されている内容は、改良のため予告無く変更する場合があります。（製品の仕様、性能、価格などはカタログ発行当時のものです）
● 本カタログに記載されている内容の一部または全部を無断で転載することは禁止されております。
● 本カタログに記載されているメーカー名、製品名などは各社の商標または登録商標です。